

# CS 477 Final Exam – Due Thursday, May 10, midnight

Three problems. Total score: 30.

## Problem 1. Verification of Sequential Programs. (10 points)

Consider the bubble-sort algorithm implemented using two nested loops in a conventional sequential language. You can assume anything you wish about the language, including support for arrays and your favorite syntax (but, obviously, no builtin support for sorting). Give the “code” for bubble-sort and prove it correct using Hoare calculus. You do not need to use ITP for this task, but make sure that your hand-crafted proof goes through all the steps. Use the Hoare rules discussed in class. If you need to extend them then do so, but say why and how.

## Problem 2. Verification of Concurrent Programs. (10 points)

1. (1 point) Informally explain the main distinction between equations and rules in rewriting logic.
2. (3 points) Give a simple concurrent program in `PARALLEL` that has a data-race; then use `search` to show that the program indeed has a data-race that leads to multiple behaviors;
3. (2 points) Add to the language `PARALLEL` discussed in class a new expression construct, `random`, that randomly evaluates to either 0 or 1;
4. (4 points) Write a program in `PARALLEL` containing a loop that randomly chooses to terminate or not. Then use Maude’s model-checker to show that the program may indeed never terminate.

## Problem 3. Runtime Verification. (10 points)

The “interval” past-time operator  $[\varphi, \psi]$  can be quite useful when specifying safety properties. Informally, its meaning is as follows: at some moment in the past  $\varphi$  was true and since then  $\psi$  has not been true. As an analogy, recall that the interval  $[0, 1)$  of real numbers contains all those points between 0 and 1, including 0 but excluding 1; therefore, being in the interval  $[0, 1)$  means that 0 has been seen in the past but 1 has not been seen yet. For example, the interval past-time operator allows you to specify properties like “ $action \rightarrow [login, logout]$ ” (actions take place only when one is logged in). Note that  $\varphi$  and  $\psi$  need not be atomic propositions in  $[\varphi, \psi]$ ; they can be any arbitrary temporal formulae, in particular ones involving intervals. This problem has three parts:

1. (2 points) Show that the interval operator is “syntactic-sugar” in past-time LTL, in the sense that it can be defined with the already given past-time LTL operators;
2. (2 points) Give a recursive semantics to the interval operator, similarly to the semantics of the other operators in Definition 1 in the lecture notes;
3. (6 points) Modify the monitor synthesis algorithm to incorporate the interval operator; you should use the recursive semantics of the interval, not its translation into the other operators. For this part, you can either modify the high-level algorithm that was described in the lecture notes or simply handle a modified working Maude code (that is, cut, paste and modify the Maude code in the lecture notes).